(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 296 224 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
26.03.2003 Bulletin 2003/13

(51) Int Cl.7: **G06F 7/72**, H04L 9/30

(21) Application number: **02015805.1**

(22) Date of filing: **15.07.2002**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **20.09.2001 JP 2001286116**

(71) Applicant: **Hitachi, Ltd.**
Chiyoda-ku, Tokyo 101-8010 (JP)

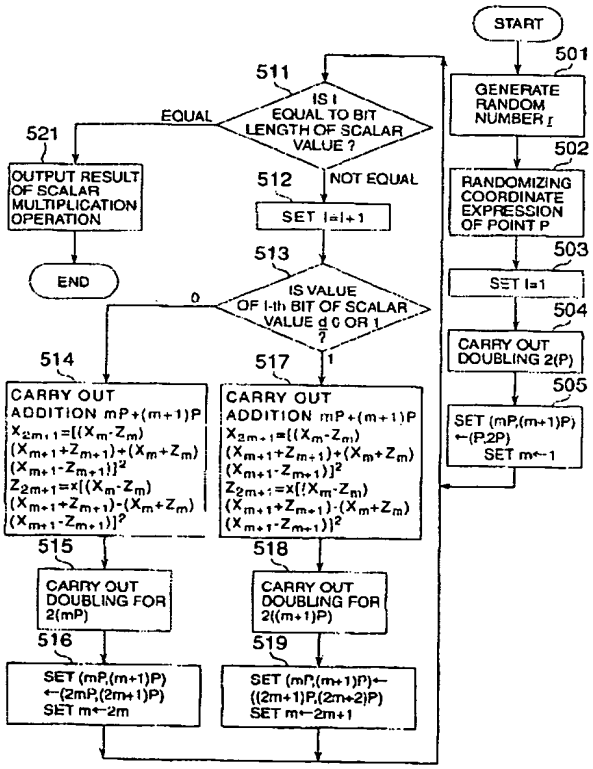
(72) Inventors:
- **Okeya, Katsuyuki, Hitachi, Ltd., Intl.Prop. Group**
  Chiyoda-ku, Tokyo 100-8220 (JP)
- **Harano, Shinichiro, Hitachi,Ltd.,**
  Intl.Prop. Group
  Chiyoda-ku, Tokyo 100-8220 (JP)

(74) Representative: **Strehl Schübel-Hopf & Partner**
Maximilianstrasse 54
80538 München (DE)

(54) Elliptic scalar multiplication system

(57) Scalar multiplication method that does not incur a higher computational cost for randomized projective coordinates of the Montgomery form of elliptic curves. Said randomized projective coordinates method is a countermeasure against side channel attacks on an elliptic curve. An attacker cannot predict the appearance of a specific value because the coordinates have been randomized.

FIG.4



EP 1 296 224 A1

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates to security technology, and particularly relates to a message processing method using an operation on an elliptic curve.

[0002] Elliptic curve cryptosystems belong to a kind of public key cryptosystem proposed by N. Koblitz and V. S. Miller. The public key cryptosystem includes information called a public key, which may be made generally open to the public, and secret information called a private key, which must be kept concealed. The public key is used for encryption or signature verification of a given message, and the private key is used for decryption or signature generation of the given message.

[0003] The private key in the elliptic curve cryptosystem is carried by a scalar value. In addition, the security of the elliptic curve cryptosystem results from difficulty in solving an elliptic curve discrete logarithm problem. The elliptic curve discrete logarithm problem means a problem of obtaining a scalar value $\underline{d}$ when there are provided a point P which is on an elliptic curve and a point dP which is a scalar multiple of the point P.

[0004] Any point on the elliptic curve designates a set of numbers satisfying a defining equation of the elliptic curve. An operation using a virtual point called the point at infinity as an identity element, that is, addition on the elliptic curve is defined all over the points on the elliptic curve. Then, addition of a point to the point itself on the elliptic curve is particularly called doubling on the elliptic curve.

[0005] Addition of two points on an elliptic curve is calculated as follows. When a straight line is drawn through the two points, the straight line intersects the elliptic curve at a third point. The point symmetric to this third intersecting point with respect to the x-axis is defined as a point resulting from the addition. For example, in the case of a Montgomery-form elliptic curve, the addition of a point (x_1, y_1) and a point (x_2, y_2), that is,

$$(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$$

is calculated and obtained by:

$$x_3 - B((y_2 - y_1) / (x_2 - x_1))^2 - A - x_1 - x_2 \quad \text{(Equation 1)}$$

$$y_3 = ((y_2 - y_1) / (x_2 - x_1))(x_1 - x_3) - y_1 \quad \text{(Equation 2)}$$

Here, A and B designates coefficients of the following defining equation of the Montgomery-form elliptic curve.

$$By^2 = x^3 + Ax^2 + x \quad \text{(Equation 3)}$$

[0006] Doubling a point on an elliptic curve is calculated as follows. When a tangent line is drawn at a point on an elliptic curve, the tangent line intersects the elliptic curve at another point. The point symmetric to this intersecting point with respect to the x-axis is defined as a point resulting from the doubling. Performing addition on a certain point a specific number of times is called scalar multiplication. The result of the scalar multiplication is called a scalar-multiplied point, and the number of times is called a scalar value.

[0007] The difficulty in solving the elliptic curve discrete logarithm problem has been established theoretically while information (computation time, power consumption and the like) involved in secret information such as a private key may leak out in the processing of encryption in real mounting. Thus, there has been proposed an attack method called side channel attack in which the secret information is recovered on the basis of the leak information.

[0008] Side channel attack on elliptic curve cryptosystems is disclosed in:

> Document 1: J. Coron, Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems, Cryptographic Hardware and Embedded Systems: Proceedings of CHES '99, LNCS 1717, Springer-Verlag, (1999) pp. 292-302.

[0009] In the elliptic curve cryptosystems, encryption, decryption, signature generation or signature verification of a given message have to be carried out with an elliptic curve operation. Particularly, calculation of scalar multiplication

on an elliptic curve is used in cryptographic processing using a scalar value as secret information.

[0010] A countermeasure against side channel attack on elliptic curve cryptosystems is disclosed in:

Document 2: K. Okeya and K. Sakurai, Power Analysis Breaks Elliptic Curve Cryptosystems even Secure Against the Timing Attack, Progress In Cryptology - INDOCRYPT 2000, LNCS 1977, Springer-Verlag, (2000), pp.178-190.

[0011] There is proposed a method using a Montgomery-form elliptic curve and randomizing points on the given elliptic curve in scalar multiplication on the elliptic curve to thereby safeguard against side channel attack.

[0012] With the development of information communication networks, cryptographic techniques have been indispensable elements for concealment or authentication about electronic information. Speeding up is demanded along with the security of the cryptographic techniques. The elliptic curve discrete logarithm problem is so difficult that elliptic curve cryptosystems can make key length shorter than that in RSA (Rivest-Shamir-Adleman) cryptosystems basing their security on the difficulty of factorization into prime factors. Thus, the elliptic curve cryptosystems open the way to comparatively high-speed cryptographic processing. However, the processing speed is not always high enough to satisfy smart cards which have restricted throughput or servers which have to carry out large volumes of cryptographic processing. It is therefore demanded to further speed up the processing in cryptosystems.

[0013] Indeed the aforementioned technique is effective as a countermeasure against side channel attack, but there is no consideration for further speeding up the processing.

SUMMARY OF THE INVENTION

[0014] It is an object of the present invention to provide an elliptic curve operation method which can safeguard against side channel attack and which is high in speed.

[0015] It is another object of the present invention to provide an encryption processing method, a decryption processing method, a signature generation method and a signature verification method using the elliptic curve operation method.

[0016] The present invention provides a scalar multiplication method for calculating a scalar-multiplied point from a scalar value and a point on an elliptic curve in the operation on the elliptic curve. The method includes the step of randomizing the point on the elliptic curve, and the step of obtaining the scalar-multiplied point of the point on the elliptic curve by the operation of a value derived from the randomized point and a value derived from the point on the elliptic curve without randomization.

[0017] The method according to the present invention may include the step of carrying out an operation upon each bit of the scalar value.

[0018] Further, according to the invention, the step of carrying out the operation upon each bit may be executed a predetermined number of times independent of the bit length of the scalar value.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019]

Fig. 1 is a system configuration diagram in an embodiment;
Fig. 2 is a sequence diagram showing delivery of information in respective embodiments;
Fig. 3 is a configuration diagram of a scalar multiplication portion in an embodiment;
Fig. 4 is a flow chart showing a first scalar multiplication method;
Fig. 5 is a flow chart showing a second scalar multiplication method;
Fig. 6 is a configuration diagram of a signature verification system in an embodiment; and
Fig. 7 is a flow chart showing a third scalar multiplication method according to a second embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS

[0020] Embodiments of the present invention will be described below with reference to the drawings.

[0021] Fig. 1 shows the configuration of a system which is connected through a network 142 and to which an elliptic curve operation method according to the present invention has been applied. In the system, a computer 101 and a computer 121 are connected through the network 142.

[0022] To encrypt a message with a public key in the computer 101 in the cryptographic communication system in Fig. 1, $P_m+k(aQ)$ and kQ are calculated and outputted.

[0023] To decrypt a cryptogram in the computer 121, it will go well if $-a(kQ)$ is calculated from the private key a and kQ, and

$$(P_m+k(aQ))-a(kQ) \quad \text{(Equation 4)}$$

is calculated and outputted. Here, P_m designates the message, k designates a random number, *a* designates a constant expressing the private key, Q designates an arbitrary base point, and *a*Q designates a point expressing the public key.

[0024] Only $P_m+k(aQ)$ and kQ are transmitted to the network 142. To recover the message P_m, it is necessary to calculate k*a*Q, that is, *a*-time multiplication of kQ. However, since the private key *a* is not transmitted to the network 142, only those who hold the private key *a* can recover the message P_m.

[0025] In Fig. 1, the computer 101 is equipped with operating units such as a CPU 113 and a coprocessor 114, storage units such as an RAM 103, an ROM 106, and an external storage unit 107, and an I/O interface 110 for carrying out data input/output with the outside of the computer. Exteriorly, there are connected a display 108, a keyboard 109, a read/write unit for portable storage media, and so on, required for a user to operate the computer 101.

[0026] Further, the computer 101 implements a storage portion 102 with the storage units such as the RAM 103, the ROM 106, and the external storage unit 107. The operating units such as the CPU 113 and the coprocessor 114 execute programs stored in the storage portion 102 so as to implement a data processing portion 112 and a scalar multiplication portion 115.

[0027] In this embodiment, the data processing portion 112 has a function as an encryption processing portion 112, encrypting an input message.

[0028] The scalar multiplication portion 115 calculates parameters required for the encryption carried out by the encryption processing portion 112. The storage portion 102 stores constants 104 (for example, a defining equation of an elliptic curve and a base point on the elliptic curve) and secret information 105 (for example, a private key), and so on.

[0029] The computer 121 has a hardware configuration similar to that of the computer 101.

[0030] Further, the computer 121 implements a storage portion 122 with storage units such as an RAM 123, an ROM 126, and an external storage unit 127. Operating units such as a CPU 133 and a coprocessor 134 execute programs stored in the storage portion 122 so as to implement a data processing portion 132 and a scalar multiplication portion 135.

[0031] In this embodiment, the data processing portion 132 has a function as a decryption processing portion 132, decrypting a cryptogram 141 which is an encrypted message.

[0032] The scalar multiplication portion 135 calculates parameters required for the decryption carried out by the decryption processing portion 132. The storage portion 122 stores constants 124 (for example, a defining equation of an elliptic curve and a base point on the elliptic curve) and secret information 125 (for example, a private key), and so on.

[0033] Fig. 2 shows the state of information delivery carried out by the respective processing portions in the computers 101 and 121.

[0034] First, description will be made on the operation in the case where the computer 101 in Fig. 1 encrypts an input message. The kind of message is no object if it is digitized data, such as text data, image data, graphic data, and audio data.

[0035] Receiving a plain message (204 in Fig. 2) through the I/O interface 110, the encryption processing portion 112 (201 in Fig. 2) judges whether the bit length of the received plane message is equal to a predetermined bit length or not. When the bit length of the plane message is longer than the predetermined length, the plane message is divided correspondingly to the predetermined bit length. Description will be made below on a partial message (also referred to as "message" simply) divided in the predetermined bit length.

[0036] Next, the encryption processing portion 112 calculates a value (y_1) of the y-coordinate of a point P_m located on an elliptic curve and having a numeric value expressed by the bit sequence of the message in an x-coordinate (x_1).

[0037] For example, a Montgomery-form elliptic curve is expressed by:

$$B(y_1)^2=(x_1)^3+A(x_1)^2+x_1 \quad \text{(Equation 5)}$$

wherein B and A are constants respectively. Accordingly, the value of the y-coordinate can be obtained therefrom.

[0038] Next, the encryption processing portion 112 generates a random number k. Then, the encryption processing portion 112 sends (206 in Fig. 2) the scalar multiplication portion 115 (202 in Fig. 2) the obtained value of the y-coordinate and the random number k together with the public key *a*Q and the x-coordinate of a point Q read (205 in Fig. 2) from the constants 104 stored in the storage portion 122 (203 in Fig. 2).

[0039] The scalar multiplication portion 115 calculates a scalar-multiplied point (x_{d1}, y_{d1})=kQ from the values of the x-coordinate and the y-coordinate of the point Q, and the random number k, and calculates a scalar-multiplied point (x_{d2}, y_{d2})=k(*a*Q) from the values of the x-coordinate and the y-coordinate of the public key *a*Q, and the random number k. The scalar multiplication portion 115 sends (207 in Fig. 2) these calculated scalar-multiplied points to the encryption

processing portion 112.

[0040] The encryption processing portion 112 carries out encryption processing using the scalar-multiplied points sent thereto. For example, for the Montgomery-form elliptic curve, $P_m+k(aQ)$ and kQ are calculated. That is, an encrypted message x_{e1}, x_{e2} is obtained by the calculation of:

$$x_{e1}=B((y_{d1}-y_1)/(x_{d1}-x_1))^2-A-x_1-x_{d1}, \quad \text{(Equation 6)}$$

$$x_{e2}-x_{d2} \quad \text{(Equation 7)}$$

[0041] The computer 101 composes (208 in Fig. 2) an encrypted output message out of at least one partial message encrypted in the encryption processing portion 112.

[0042] The computer 101 outputs the encrypted output message as data 141 through the I/O interface 110, and transfers the data 141 to the computer 121 through the network 142.

[0043] Incidentally, reading information from the storage portion 203 in Fig. 2 may be performed before the acceptance of the input message.

[0044] Next, description will be made on the operation when the computer 121 decrypts the encrypted message 141, with reference to Fig. 2.

[0045] Supplied with the encrypted data 141 (input message 204 in Fig. 2) through the I/O interface 110, the decryption processing portion 132 (data processing portion 201 in Fig. 2) judges whether the bit length of the supplied encrypted data 141 is equal to a predetermined bit length or not. When the bit length of the data 141 is longer than the predetermined length, the encrypted data is divided correspondingly to the predetermined bit length. Description will be made below on partial data (also referred to as "data" simply) divided in the predetermined bit length.

[0046] A value of the y-coordinate of a point located on an elliptic curve and having a numeric value expressed by the bit sequence of the data 141 in the x-coordinate is calculated.

[0047] On the assumption that the encrypted message is of a bit sequence of x_{e1}, x_{e2}, and the curve is a Montgomery-form elliptic curve, the value (y_{e1}) of the y-coordinate can be obtained by:

$$B(y_{e1})^2=(x_{e1})^3+A(x_{e1})^2+x_{e1} \quad \text{(Equation 8)}$$

(wherein B and A are constants respectively).

[0048] The decryption processing portion 132 reads (205 in Fig. 2) the private key ***a*** from the secret information 125 stored in the storage portion 122 (203 in Fig. 2), and sends (206 in Fig. 2) the private key ***a*** together with the values (x_{e1}, y_{e1}) of the x-coordinate and the y-coordinate to the scalar multiplication portion 135 (202 in Fig. 2).

[0049] The scalar multiplication portion 135 calculates a scalar-multiplied point $(x_{d3}, y_{d3})=a(x_{e2}, y_{e2})$ from the values of the x-coordinate and the y-coordinate, and the private key a of the secret information 125.

[0050] The scalar multiplication portion 135 sends (207 in Fig. 2) the calculated scalar-multiplied point to the decryption processing portion 132. The decryption processing portion 132 carries out decryption processing using the scalar-multiplied point sent thereto.

[0051] For example, when the encrypted message is of a bit sequence of X_{e1}, X_{e2}, and the curve is a Montgomery-form elliptic curve, the decryption processing is attained by the calculation of:

$$(P_m+k(aQ))-a(kQ)=(x_{e1}, y_{e1})-(x_{d3}, y_{d3})$$

That is, x_{f1} corresponding to the partial message x_1 which has not yet been encrypted is obtained by the calculation of:

$$x_{f1}=B((y_{e1}+y_{d3}) / (x_{e1}-x_{d3}))^2-A-x_{e1}-x_{d3} \quad \text{(Equation 9)}$$

[0052] The computer 121 composes (208 in Fig. 2) a plane message out of such partial messages decrypted by the decryption processing portion 132. The computer 121 outputs the plane message from the display 108 or the like through the I/O interface 110.

[0053] Next, description will be made on the details of the processing of the scalar multiplication portion 135 when the computer 121 performs the decryption processing.

[0054] Fig. 3 shows functional blocks of a scalar multiplication portion used in respective embodiments. The scalar multiplication portion 202 is constituted by a randomizing portion 402, an adding portion 403, a doubling portion 404, a bit value judging portion 405, and a repetition judging portion 406.

[0055] A method (referred to as "first calculation method") in which the scalar multiplication portion 202 calculates a scalar-multiplied point dP on a Montgomery-form elliptic curve from a scalar value d and a point P on the Montgomery-form elliptic curve will be described with reference to Fig. 4. Consider message-related data expressed as a point on the elliptic curve.

[0056] When the scalar multiplication portion 202 receives the scalar value d and the point P on the elliptic curve from the decryption processing portion 132, the randomizing portion 402 randomizes the received point P on the elliptic curve. This is attained by the following processing carried out by the randomizing portion 402.

[0057] A random number r is generated (501).

[0058] The point P=(x, y) is expressed (502) as a randomized point P=(rx, ry, r) in projective coordinates. Here, $r \neq 0$.

[0059] The initial value 1 is substituted (503) for a variable I.

[0060] The doubling portion 404 calculates (504) a doubled point 2P of the randomized point P by use of doubling formulae in the projective coordinates on the Montgomery-form elliptic curve.

[0061] The doubling formulae in the projective coordinates on the Montgomery-form elliptic curve include:

$$4X_1Z_1=(X_1+Z_1)^2-(X_1-Z_1)^2 \quad \text{(Equation 10)}$$

$$X_2=(X_1+Z_1)^2(X_1-Z_1)^2 \quad \text{(Equation 11)}$$

$$Z_2=(4X_1Z_1)((X_1-Z_1)^2+((A+2)/4)\ (4X_1Z_1)) \quad \text{(Equation 12)}$$

wherein A designates a constant, X_1, Z_1, X_2 and Z_2 designate the X-coordinate and the Z-coordinate of the point P, and the X-coordinate and the Z-coordinate of the point 2P, respectively.

[0062] The set of points (P, 2P) made of the randomized point P and the point 2P obtained in Step 504 are stored (505) temporarily as a set of points (mP, (m+1)P) (m is a natural number) at m=1 into the storage portion 122.

[0063] The repetition judging portion 406 judges whether the variable I coincides with the bit length of the scalar value d read from the storage portion 122 or not.

[0064] When they coincide with each other, the routine of processing goes to Step 521. On the other hand, when they do not coincide with each other, the routine of processing goes to Step 512 (511). When they do not coincide with each other in Step 511, the variable I is increased by 1 (512).

[0065] The bit value judging portion 405 judges whether the value of the I-th bit of the scalar value d is 0 or 1. When the value is 0, the routine of processing goes to Step 514. When the value is 1, the routine of processing goes to Step 517 (513).

[0066] When the value of the bit is 0 in Step 513, the adding portion 403 carries out addition mP+(m+1)P of the point P and the point (m+1)P from the set of points (mP, (m+1)P) expressed in the projective coordinates by use of the point P=(x, y) which has not been randomized. Thus, the point (2m+1)P is calculated (514).

[0067] This is attained by the calculation of:

$$X_{2m+1}=[(X_m-Z_m)\ (X_{m+1}+Z_{m+1})+(X_m+Z_m)\ (X_{m+1}-Z_{m+1})]^2, \quad \text{(Equation 13)}$$

$$Z_{2m+1}=x[(X_m-Z_m)\ (X_{m+1}+Z_{m+1})-(X_m+Z_m)\ (X_{m+1}-Z_{m+1})]^2 \quad \text{(Equation 14)}$$

Here, X_m, Z_m, X_{m+1}, Z_{m+1}, X_{2m+1} and Z_{2m+1} designate the X-coordinate and the Z-coordinate of the point mP, the X-coordinate and the Z-coordinate of the point (m+1)P, and the X-coordinate and the Z-coordinate of the point (2m+1)P, respectively.

[0068] The doubling portion 404 performs an addition on the elliptic curve, namely doubling 2(mP) of the point mP from the set of points (mP, (m+1)P) expressed in the projective coordinates, so as to calculate the point 2mP (515). This is attained by the calculation of:

$$4X_mZ_m = (X_m+Z_m)^2 - (X_m-Z_m)^2 \quad \text{(Equation 15)}$$

$$X_{2m} = (X_m+Z_m)^2(X_m-Z_m)^2 \quad \text{(Equation 16)}$$

$$Z_{2m} = (4X_mZ_m)\,((X_m-Z_m)^2+((A+2)/4)(4X_mZ_m)) \quad \text{(Equation 17)}$$

Here, A designates a constant, and X_m, Z_m, X_{2m} and Z_{2m} designate the X-coordinate and the Z-coordinate of the point mP, and the X-coordinate and the Z-coordinate of the point 2mP, respectively.

[0069] The set of points (mP, (m+1)P) is replaced by the set of points (2mP, (2m+1)P) made of the point 2mP obtained in Step 515 and the point (2m+1)P obtained in Step 514, and 2m is substituted for m. Then, the routine of processing returns to Step 511 (516).

[0070] When the value of the bit is 1 in Step 513, the adding portion 403 carries out addition mP+(m+1)P of the point mP and the point (m+1)P from the set of points (mP, (m+1)P) expressed in the projective coordinates by use of the point P=(x, y) which has not been randomized. Thus, the point (2m+1)P is calculated (517).

[0071] This is attained by the calculation of:

$$X_{2m+1} = [(X_m-Z_m)\,(X_{m+1}+Z_{m+1}) + (X_m+Z_m)\,(X_{m+1}-Z_{m+1})]^2 \quad \text{(Equation 18)}$$

$$Z_{2m+1} = x[(X_m-Z_m)\,(X_{m+1}+Z_{m+1}) - (X_m+Z_m)\,(X_{m+1}-Z_{m+1})]^2 \quad \text{(Equation 19)}$$

[0072] The doubling portion 404 performs an addition on the elliptic curve, namely doubling 2((m+1)P) of the point (m+1)P from the set of points (mP, (m+1)P) expressed in the projective coordinates, so as to calculate the point (2m+2)P (518).

This is attained by the calculation of:

$$4X_{m+1}Z_{m+1} = (X_{m+1}+Z_{m+1})^2 - (X_{m+1}-Z_{m+1})^2 \quad \text{(Equation 20)}$$

$$X_{2m+2} = (X_{m+1}+Z_{m+1})^2(X_{m+1}-Z_{m+1})^2 \quad \text{(Equation 21)}$$

$$Z_{2m+2} =$$

$$(4X_{m+1}Z_{m+1})\,((X_{m+1}-Z_{m+1})^2+((A+2)/4)(4X_{m+1}Z_{m+1})) \quad \text{(Equation 22)}$$

Here, A designates a constant, and X_{m+1}, Z_{m+1}, X_{2m+2} and Z_{2m+2} designate the X-coordinate and the Z-coordinate of the point (m+1)P, and the X-coordinate and the Z-coordinate of the point (2m+2)P, respectively.

[0073] The set of points (mP, (m+1)P) is replaced by the set of points ((2m+1)P, (2m+2)P) made of the point (2m+1)P obtained in Step 517 and the point (2m+2)P obtained in Step 518, and 2m+1 is substituted for m. Then, the routine of processing returns to Step 511 (519).

[0074] When the variable I coincides with the bit length of the scalar value d in Step 511, the values X_m and Z_m are obtained as the X-coordinate and the Z-coordinate of the scalar-multiplied point dP from the point $mP=(X_m, Y_m, Z_m)$ expressed in the projective coordinates from the set of points (mP, (m+1)P) expressed in the projective coordinates. The obtained values X_m and Z_m are outputted as the scalar-multiplied point dP to the decryption processing portion 132 (521).

[0075] Here, the Y-coordinate may be obtained in an Y-coordinate recovery method, and outputted together, or the coordinates transformed into affine coordinates or the like may be outputted. Alternatively, the coordinates transformed into coordinates on a Weierstrass-form elliptic curve may be outputted.

[0076] The Y-coordinate recovery method is disclosed in:

Document 3: K. Okeya and K. Sakurai, Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm

with Recovery of the y-Coordinate on a Montgomery-Form Elliptic Curve, Cryptographic Hardware and Embedded Systems: Proceedings of CHES 2001, (2001) pp.129-144.

[0077] In the above procedure, the value m and the scalar value d have equal bit length and the same bit pattern. Thus, the values are equal to each other. This means that the calculation of the scalar-multiplied point dP is completed in the above procedure.

[0078] Incidentally, although the point on the elliptic curve to be supplied to the scalar multiplication portion 202 is set as a point on a Montgomery-form elliptic curve, it may be a point on a Weierstrass-form elliptic curve. In this case, it will go well if the point on the Weierstrass-form elliptic curve transformed into a point on a Montgomery-form elliptic curve is used.

[0079] The computational cost of the operation of addition in the projective coordinates on the Montgomery-form elliptic curve in Step 514 and Step 517 is 3M+2S when the computational cost of multiplication on a finite field is M and the computational cost of squaring on a finite field is S. This computational cost is equal to that when randomization is not carried out on the point P in Step 502.

[0080] If the operation of addition is calculated with the randomized point P in Step 514 and Step 517, the computational cost will reach 4M+2S, increasing by M in comparison with that in the aforementioned algorithm using the point P not randomized.

[0081] The number of times of repetition of Step 511 to Step 519 is (bit length of scalar value d)-1 times. The total computational cost in the aforementioned algorithm is smaller by (k-1)M than that in the algorithm using the randomized point P in Step 514 and Step 517. Thus, the processing speed is higher so much. Here, k designates the bit length of the scalar value d.

[0082] In addition, the aforementioned method is also effective as a countermeasure against side channel attack. This reason is as follows.

[0083] The point P randomized in Step 502 is used in the following steps.

[0084] In Step 514 and Step 517, the point P not randomized is used. However, in Step 514 and Step 517, the operation for calculating the point (2m+1)P is performed by use of the points mP and (m+1)P derived from the randomized point P, and the point P not randomized. If another value is generated in Step 501 for generating a random number so that the values of the coordinates of the point P randomized in Step 502 are varied, the values of the coordinates of the points mP and (m+1)P will be varied in Step 514 and Step 517. Thus, the values of the coordinates of the point (2m+1)P calculated by use of those values will be varied. That is, even if the same scalar value d and the same point P are provided, the values of the coordinates of the point (2m+1)P will be varied whenever they are calculated.

[0085] Further, the same procedure of computations is carried out regardless of the result of judgement about the value of the bit in Step 513. It is therefore proved that there is no dependency relation between the execution sequence of computations and the value of the bit.

[0086] When this calculation method is mounted, the same program or processing circuit may be formed to be shared regardless of the bit value, with respect to the processings in Step 513 et seq.

[0087] As described above, the first calculation method provides no information useful to side channel attack. Thus, the method is immune to side channel attack. In addition, the method has a feature in that calculation can be performed at a high speed in accordance with the properties of the elliptic curve used therein.

[0088] Next, a method (referred to as "second calculation method") in which the scalar multiplication portion 202 calculates a scalar-multiplied point dP on a Weierstrass-form elliptic curve from a scalar value d and a point P on the Weierstrass-form elliptic curve will be described with reference to Fig. 5.

[0089] When the scalar multiplication portion 202 receives the point P on the elliptic curve and the scalar value d from the decryption processing portion 132, the randomizing portion 402 randomizes the received point P on the elliptic curve. This is attained by the following processing carried out by the randomizing portion 402.

[0090] A random number r is generated (601).

[0091] The point P=(x, y) is expressed as (r^2x, r^3y, r) in Jacobian coordinates (602). Here, r, r^2 and $r^3 \neq 0$, expressing the degrees of weighting.

[0092] Next, the initial value 1 is substituted for a variable I (603).

[0093] The point P randomized in Step 602 is stored temporarily as a point R into the storage portion 122 (604).

[0094] The repetition judging portion 406 judges whether the variable I coincides with the bit length of the scalar value d or not.

[0095] When they coincide with each other, the routine of processing goes to Step 621. On the other hand, when they do not coincide with each other, the routine of processing goes to Step 612 (611).

[0096] When they do not coincide with each other in Step 611, the variable I is increased by 1 (612).

[0097] The doubling portion 404 carries out doubling 2(R) of the point R expressed in the Jacobian coordinates, and stores the point 2R into Q[0] (613).

[0098] The adding portion 403 carries out addition Q[0]+P of the point Q[0] expressed in the Jacobian coordinates, and the point P=(x, y) not randomized, and stores the result of the addition into Q[1] (614).

[0099] The bit value judging portion 405 judges whether the value of the i-th bit of the scalar value d is 0 or 1. When the value is 0, the routine of processing goes to Step 616. When the value is 1, the routine of processing goes to Step 617 (615).

[0100] When the value of the bit is 0 in Step 615, the point Q[0] obtained in Step 613 is stored as the point R, and the routine of processing returns to Step 611 (616).

[0101] When the value of the bit is 1 in Step 615, the point Q[1] obtained in Step 614 is stored as the point R, and the routine of processing returns to Step 611 (617).

[0102] When the variable i coincides with the bit length of the scalar value $\underline{d}$ in Step 611, the point R expressed in the Jacobian coordinates is outputted as the scalar-multiplied point dP to the decryption processing portion 132 (621).

[0103] Here, the point transformed into affine coordinates or the like may be outputted. Alternatively, the point transformed into coordinates on a Montgomery-form elliptic curve may be outputted. Incidentally, although the point on the elliptic curve to be supplied to the scalar multiplication portion 202 is set as a point on a Weierstrass-form elliptic curve, it may be a point on a Montgomery-form elliptic curve. In this case, it will go well if the point on the Montgomery-form elliptic curve transformed into a point on a Weierstrass-form elliptic curve is used.

[0104] The computational cost of the operation of addition in the Jacobian coordinates on the Weierstrass-form elliptic curve in Step 614 is 8M+3S. This computational cost is equal to that when randomization is not carried out on the point P in Step 602. If the operation of addition is calculated with the randomized point P in Step 614, the computational cost of the operation will reach 12M+4S, increasing by 4M+S in comparison with that in the aforementioned algorithm using the point P not randomized. The number of times of repetition of Step 611 to Step 617 is (bit length of scalar value $\underline{d}$)-1 times. The total computational cost in the aforementioned algorithm is smaller by (k-1)(4M+S) than that in the algorithm using the randomized point P in Step 614. Thus, the processing speed is higher so much. Here, k designates the bit length of the scalar value $\underline{d}$.

[0105] In addition, the aforementioned method is also effective as a countermeasure against side channel attack. This reason is as follows.

[0106] The point P randomized in Step 602 is used in the following steps.

[0107] In Step 614, the point P not randomized is used. However, the operation Q[0]+P is calculated by use of the point Q[0] derived from the randomized point P, and the point P not randomized. If another value is generated in Step 601 for generating a random number so that the values of the coordinates of the point P randomized therewith in Step 602 are varied, the values of the coordinates of the point Q[0] in Step 614 will be varied, and hence the values of the coordinates of the point Q[0]+P calculated with the varied values will be varied. That is, even if the same scalar value $\underline{d}$ and the same point P are provided, the values of the coordinates of the point Q[0] will be varied whenever they are calculated.

[0108] Further, the same procedure of computations is carried out regardless of the result of judgement on the value of the bit in Step 615. Accordingly, there is no dependency relation between the execution sequence of computations and the value of the bit. Thus, the aforementioned algorithm is immune to side channel attack.

[0109] As described above, the aforementioned method provides no information useful to side channel attack. Thus, the method is immune to side channel attack. In addition, the second calculation method has a feature in that it is applicable to elliptic curves used generally, in comparison with the first calculation method.

[0110] Incidentally, although the Weierstrass-form elliptic curve is used as the elliptic curve in the second calculation method, an elliptic curve defined on a finite field of characteristics 2 may be used, or an elliptic curve defined on an OEF (Optimal Extension Field) may be used.

[0111] There is a statement about OEFs in:

> Document 4: D. V. Bailey and C. Paar, Optimal Extension Fields for Fast Arithmetic in Public-key Algorithms, Advances in Cryptology CRYPTO '98, LNCS1462, (1998), pp.472-485.

[0112] Although description has been made above on the operation of the scalar multiplication portion 135 in the case where the computer 121 has decrypted the encrypted data 141, similar things can be applied to the case where the computer 101 encrypts an input message.

[0113] In that case, the scalar multiplication portion 115 of the computer 101 outputs the point Q on the elliptic curve, the scalar-multiplied point kQ using the random number $\underline{k}$, and the scalar-multiplied point k(aQ) using the public key aQ and the random number $\underline{k}$, which have been already described. At this time, the respective scalar-multiplied points can be obtained in similar processings carried out with the random number $\underline{k}$ substituted for the scalar value $\underline{d}$ described in the first and second calculation methods, with the point Q on the elliptic curve and the public key aQ substituted for the point P on the elliptic curve described in the first and second calculation methods, and with aQ as the public key.

[0114] Next, a method (referred to as "third calculation method") in which the scalar multiplication portion 202 cal-

culates a scalar-multiplied point dP on a Montgomery-form elliptic curve from a scalar value $\underline{d}$ of actual bit length L and a point P on the Montgomery-form elliptic curve will be described with reference to Fig. 7. Here, the actual bit length means the number of bits of the area (such as a memory or a register) where the scalar value $\underline{d}$ is stored. Therefore, the most significant bit does not have to be 1.

[0115] This method is designed so that the computation steps and the computation time are fixed regardless of the scalar value $\underline{d}$. Accordingly, the method provides no information useful to the aforementioned method of attack. Thus, the method is immune thereto.

[0116] Receiving the point P on the elliptic curve and the scalar value d from the decryption processing portion 132, the scalar multiplication portion 202 judges whether the scalar value $\underline{d}$ is 0 or not. When the scalar value $\underline{d}$ is 0, the scalar multiplication portion 202 outputs the point at infinity, and then terminates the processing. When the scalar value $\underline{d}$ is not 0, the scalar multiplication portion 202 keeps on with the processing (1201).

[0117] The randomizing portion 402 randomizes the received point P on the elliptic curve. That is:

A random number $\underline{r}$ is generated (1202).

[0118] The point P is expressed as (rx, ry, r) in projective coordinates (1203).

[0119] Next, indefinite points $T_{0,0}$, $T_{0,1}$, $T_{1,0}$ and $T_{1,1}$ on the elliptic curve are initialized. The point P randomized in Step 1203, the indefinite point $T_{0,0}$, the doubled point 2P of the point P randomized in Step 1203, and the indefinite point $T_{0,1}$ are substituted for the indefinite points $T_{0,0}$, $T_{0,1}$, $T_{1,0}$ and $T_{1,1}$ respectively. The doubled point 2P of the randomized point P is calculated by use of the doubling formulae (Equations 10, 11 and 12) in the projective coordinates on the Montgomery-form elliptic curve (1204).

[0120] The initial value 0 is substituted for a variable $\underline{s}$ (1205).

[0121] The initial value L-1 is substituted for a variable $\underline{i}$ (1206).

[0122] The repetition judging portion 406 judges whether the variable $\underline{i}$ is smaller than 0 or not. When the variable $\underline{i}$ is not smaller than 0, the routine of processing goes to Step 1208. When the variable $\underline{i}$ is smaller than 0, the routine of processing goes to Step 1213 (1207).

[0123] A point $T_{s,di}$ is substituted for an indefinite point T on the elliptic curve. The value d_i corresponds to a bit d_i at j=i on the expression that the scalar value $d=\Sigma d_j 2^j, d_j \in \{0,1\}, j$ moves between 0 and L-1 (1208).

[0124] The doubling portion 404 carries out doubling 2(T) of the point T expressed in projective coordinates, and stores the obtained point 2T into the point $T_{s,di}$ (1209).

[0125] The adding portion 403 carries out addition of the point T expressed in the projective coordinates and the point $T_{s,1-di}$ expressed in the projective coordinates by use of the point T=(x, y) not randomized, and stores the result of the addition into the point $T_{s,1-di}$ (1210).

[0126] Logical sum of $\underline{s}$ and d_i is performed, and the result of the logical sum is stored into $\underline{s}$ (1211).

[0127] The variable $\underline{i}$ is decreased by 1 (1212).

[0128] When i<0 in Step 1207, the point $T_{1,0}$ expressed in the projective coordinates is outputted as the scalar-multiplied point dP to the decryption processing portion 132 (1213).

[0129] Here, the Y-coordinate may be obtained in an Y-coordinate recovery method, and outputted together, or the coordinates transformed into affine coordinates or the like may be outputted. Alternatively, the coordinates transformed into coordinates on a Weierstrass-form elliptic curve may be outputted. There is a statement about the Y-coordinate recovery method in Document 3.

[0130] Incidentally, although the point on the elliptic curve to be supplied to the scalar multiplication portion 202 is set as a point on a Montgomery-form elliptic curve, it may be a point on a Weierstrass-form elliptic curve. In this case, it will go well if the point on the Weierstrass-form elliptic curve transformed into a point on a Montgomery-form elliptic curve is used.

[0131] The computational cost of the operation of addition in the projective coordinates on the Montgomery-form elliptic curve in Step 1210 is 3M+2S. This computational cost is equal to that when randomization is not carried out on the point P in Step 1203.

[0132] If the operation of addition is calculated with the randomized point P in Step 1210, the computational cost of the operation will reach 4M+2S, increasing by M in comparison with that in the aforementioned algorithm using the point P not randomized.

[0133] The number of times of repetition of Step 1207 to Step 1212 is L times. The total computational cost in the aforementioned algorithm is smaller by LM than that in the algorithm using the randomized point P in Step 1210. Thus, the processing speed is higher so much.

[0134] In addition, the aforementioned third calculation method is also effective as a countermeasure against side channel attack. This reason is as follows.

[0135] The point P randomized in Step 1203 is used in the following steps.

[0136] In Step 1210, the point P not randomized is used. However, in Step 1210, the point $T+T_{s,1-di}$ is calculated by

use of the points T and $T_{s,1-di}$ derived from the randomized point P, and the point P not randomized. If another value is generated in Step 1202 for generating a random value so that the values of the coordinates of the point P randomized in Step 1203 are varied, the values of the coordinates of the points T and $T_{s,1-di}$ will be varied in Step 1210. Thus, the values of the coordinates of the point $T+T_{s,1-di}$ calculated by use of those values will be varied. That is, even if the same scalar value $\underline{d}$ and the same point P are provided, the values of the coordinates of the point $T+T_{s,1-di}$ will be varied whenever they are calculated.

[0137] Further, the same procedure of computations is carried out regardless of the value of each bit d_i. Accordingly, there is no dependency relation between the execution sequence of computations and the value of the bit.

[0138] In addition, the number of times of repetition of the Step 1207 to Step 1212 does not depend on the bit length of the value $\underline{d}$, but always takes L times. Thus, the execution sequence of computations does not depend on the bit length of the value $\underline{d}$, either.

[0139] Incidentally, the bit d_{L-1} may be substituted for $\underline{s}$ in Step 1205, and L-2 be substituted for I in Step 1206. In this case, there is produced no dummy operation when the most significant bit d_{L-1} of the scalar value $\underline{d}$ is 1. That is, the initial repetition of Steps 1207-1212 carried out when s=0 and i=L-1 can be omitted so that the algorithm can be further speeded up.

[0140] As described above, the third calculation method provides no information useful to side channel attack. Thus, the method is immune to side channel attack.

[0141] Next, an embodiment in which the present invention is applied to a signature verification system will be described with reference to Fig. 6 and Fig. 2.

[0142] The signature verification system in Fig. 6 is constituted by a smart card 701 and a computer 721 for performing signature verification processing.

[0143] In terms of functions, the smart card 701 has a configuration similar to that of the computer 101. Not the encryption processing portion 112 but a signature generation processing portion 712 for providing message data or a signature is implemented with operating units such as a CPU 733 and a coprocessor 734, and programs stored in a storage portion 722. Incidentally, there is not provided any external storage unit, any display, or any keyboard.

[0144] The computer 721 has a configuration similar to that of the computer 101, and not the decryption processing portion 132 but a signature verification processing portion 732 is implemented with a CPU 733 and programs.

[0145] Scalar multiplication portions 715 and 735 have functions similar to those of the scalar multiplication portions 115 and 135 shown in Fig. 1, respectively.

[0146] The operation of signature generation and signature verification in the signature verification system in Fig. 6 will be described with reference to Fig. 2.

[0147] The computer 721 transmits a numeric value selected at random as a challenge code 743 to the smart card 701.

[0148] The signature generation processing portion 712 (201 in Fig. 2) accepts the challenge code 743, gets the hash value of the challenge code 743, and transforms the hash value into a numeric value $\underline{f}$ of predetermined bit length.

[0149] The signature generation processing portion 712 generates a random number $\underline{u}$, and sends (206 in Fig. 2) the random number $\underline{u}$ to the scalar multiplication portion 715 (202 in Fig. 2) together with a base point Q on the elliptic curve read (205 in Fig. 2) from constants 704 stored in the storage portion 702 (203 in Fig. 2).

[0150] The scalar multiplication portion 715 calculates a scalar-multiplied point (x_u, y_u) using the base point Q and the random number $\underline{u}$, and sends (207 in Fig. 2) the calculated scalar-multiplied point to the signature generation processing portion 712.

[0151] The signature generation processing portion 712 generates a signature by use of the scalar-multiplied point sent thereto. For example, in the case of an ECDSA signature, a signature (s, t) corresponding to the challenge code 743 is obtained (208 in Fig. 2) by the calculation of:

$$s=x_u \bmod q \qquad \text{(Equation 23)}$$

$$t=u^{-1}(f+ds) \bmod q \qquad \text{(Equation 24)}$$

[0152] Here, the value $\underline{q}$ designates the order of the base point Q, that is, such a numeric value that the q-multiplied point qQ of the base point Q becomes the point at infinity while an m-multiplied point mQ of the base point Q with respect to a numeric value $\underline{m}$ smaller than the value $\underline{q}$ is not the point at infinity.

[0153] There is a statement about the ECDSA signature in:

Document 5: ANSI X9.62 Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital

Signature Algorithm (ECDSA), (1999).

[0154] The smart card 701 outputs the signature 741 generated in the signature generation processing portion 712 through an I/O interface 710. The signature 741 is transferred to the computer 721.

[0155] Receiving (204 in Fig. 2) the signature 741, the signature verification processing portion 732 (201 in Fig. 2) of the computer 721 examines whether the numeric values $\underline{s}$ and $\underline{t}$ of the signature 741 are within a suitable range, that is, satisfy $1 \leq s, t < q$.

[0156] When the numeric values $\underline{s}$ and $\underline{t}$ are not within the aforementioned range, the signature verification processing portion 732 outputs "invalid" as the result of signature verification for the challenge code 743, and rejects the smart card 701. When the numeric values $\underline{s}$ and $\underline{t}$ are within the aforementioned range, the signature verification processing portion 732 performs the calculation of:

$$h=t^{-1} \bmod q \qquad \text{(Equation 25)}$$

$$h_1=fh \bmod q \qquad \text{(Equation 26)}$$

$$h_2=sh \bmod q \qquad \text{(Equation 27)}$$

Then, the signature verification processing portion 732 sends (206 in Fig. 6) the scalar multiplication portion 735 (202 in Fig. 2) the calculated values h_1 and h_2 together with a public key aQ and the base point Q read (205 in Fig. 2) from the constants 724 stored in the storage portion 722 (203 in Fig. 2).

[0157] The scalar multiplication portion 735 calculates a scalar-multiplied point h_1Q using the base point Q and the value h_1 and a scalar-multiplied point h_2aQ using the public key aQ and the value h_2, and sends (207 in Fig. 2) the calculated scalar-multiplied points to the signature verification processing portion 732.

[0158] The signature verification processing portion 732 performs signature verification processing using the scalar-multiplied points sent thereto. For example, a point R is calculated by:

$$R=h_1Q+h_2aQ \qquad \text{(Equation 28)}$$

[0159] When the x-coordinate of the point R is x_R, a value s' is calculated by:

$$s'=x_R \bmod q \qquad \text{(Equation 29)}$$

[0160] When s'=s, the signature verification processing portion 732 outputs "valid" as the result of signature verification for the challenge code 743, authenticates and accepts (208 in Fig. 2) the smart card 701.

[0161] When not s'=s, the signature verification processing portion 732 outputs "invalid", and rejects (208 in Fig. 2) the smart card.

[0162] The scalar multiplication portion 715 or 735 in the above embodiment has a function similar to that of the scalar multiplication portion 115 or 135 in Fig. 1. Accordingly, scalar multiplication can be performed at high speed while safeguarding against side channel attack.

[0163] Accordingly, the smart card 701 engaging in signature generation processing and the computer 721 engaging in signature verification processing can safeguard against side channel attack and further carry out the processing at high speed.

[0164] Next, an embodiment in which the present invention is applied to a key exchange system will be described. In this embodiment, the system configuration of Fig. 1 can be applied.

[0165] The data processing portions 112 and 132 in Fig. 1 function as key exchange processing portions 112 and 132 in this embodiment, respectively.

[0166] The operation in the case where the computer 101 in the key exchange system derives shared information from input data 143 will be described with reference to Figs. 1 and 2.

[0167] The data processing portion 132 (201 in Fig. 2) of the computer 121 reads a secret key b from the constants 124 in the storage portion 122 (203 in Fig. 2), and calculates a public key bQ of the computer 121. Then, the public key bQ is transferred as data 143 to the computer 101 through the network 142.

[0168] When the key exchange processing portion 112 (201 in Fig. 2) of the computer 101 accepts (204 in Fig. 2) the input of the public key ***b*Q** of the computer 121, the key exchange processing portion 112 sends (206 in Fig. 2) the scalar multiplication portion 115 (202 in Fig. 2) the public key ***b*Q** of the computer 121 together with a private key ***a*** of the computer 101 which is secret information 105 read (205 in Fig. 2) from the storage portion 102 (203 in Fig. 2).

[0169] The scalar multiplication portion 115 calculates a scalar-multiplied point ***ab*Q** using the private key ***a*** and the public key ***b*Q**, and sends (207 in Fig. 2) the calculated scalar-multiplied point to the key exchange processing portion 112.

[0170] The key exchange processing portion 112 derives shared information by use of the scalar-multiplied point sent thereto, and stores the derived shared information as secret information 105 into the storage portion 102. For example, the x-coordinate of the scalar-multiplied point ***ab*Q** is set as shared information.

[0171] Next, description will be made on the operation when the computer 121 derives the shared information from the input data 141.

[0172] The data processing portion 112 (201 in Fig. 2) of the computer 101 reads a secret key ***a*** from the constants 104 in the storage portion 102 (203 in Fig. 2), and calculates a public key ***a*Q** of the computer 101. Then, the public key ***a*Q** is transferred as data 141 to the computer 121 through the network 142.

[0173] When the key exchange processing portion 132 (201 in Fig. 2) of the computer 121 accepts (204 in Fig. 2) the input of the public key ***a*Q** of the computer 101, the key exchange processing portion 132 sends (206 in Fig. 2) the scalar multiplication portion 135 (202 in Fig. 2) the public key ***a*Q** of the computer 101 together with a private key ***b*** of the computer 121 which is secret information 125 read (205 in Fig. 2) from the constants 124 in the storage portion 122.

[0174] The scalar multiplication portion 135 calculates a scalar-multiplied point ***ba*Q** using the private key ***b*** and the public key ***a*Q**, and sends (207 in Fig. 2) the calculated scalar-multiplied point to the key exchange processing portion 132.

[0175] The key exchange processing portion 132 derives shared information by use of the scalar-multiplied point sent thereto, and stores the derived shared information as secret information 125 into the storage portion 122. For example, the x-coordinate of the scalar-multiplied point ***ba*Q** is set as shared information.

[0176] Here, since the number ***ab*** and the number ***ba*** are identical as numeric value, the point ***ab*Q** and the point ***ba*Q** indicate the same point, resulting in the derivation of the same information.

[0177] Although the point ***a*Q** and the point ***b*Q** are transmitted onto the network 142, the private key ***a*** or the private key ***b*** has to be used to calculate the point ***ab*Q** (or the point ***ba*Q**). That is, those who do not know the private key **a** or the private key ***b*** cannot obtain the shared information. The shared information obtained thus can be utilized as a private key in a private key cryptosystem.

[0178] Also in this embodiment, since the scalar multiplication portions 115 and 135 have the aforementioned features, they can perform key exchange processing at high speed while safeguarding against side channel attack.

[0179] In addition, the encryption processing portion, the decryption processing portion, the signature generation portion, the signature verification portion and the key exchange processing portion in the above description may be implemented with special hardware. In addition, the scalar multiplication portion may be implemented with a coprocessor or other special hardware.

[0180] In addition, the data processing portion may be designed to be able to perform at least one processing of the encryption processing, the decryption processing, the signature generation processing, the signature verification processing and the key exchange processing described previously.

[0181] It should be further understood by those skilled in the art that although the foregoing description has been made on embodiments of the invention, the invention is not limited thereto and various changes and modifications may be made without departing from the spirit of the invention and the scope of the appended claims.

Claims

1. A scalar multiplication method for calculating data for encrypting a message in an information processing system, comprising:

   a first step of randomizing (502, 602, 1203) message-related data expressed as points on an elliptic curve; and a second step of operating (504, 514, 515, 517, 518, 613, 614, 1204, 1209, 1210) a first value derived from said randomized points, and a second value derived from said points on said elliptic curve without randomization of said message-related data.

2. The method of claim 1, wherein said first value and said second value are added (514, 517, 614, 1210).

3. The method of claim 1, wherein said second step is repeated in a loop.

4. The method of claim 2, said second step further including the step of adding (515, 518) a result obtained by adding said first value and said second value on an elliptic curve.

5. The method of claim 2, further comprising the steps of:

adding (504) said first value on an elliptic curve; and
adding (515, 518) a result obtained by adding said second value and a result of adding said first value on an elliptic curve.

6. The method of claim 1, said first step further including the step of transforming (502, 1203) a coordinate system expressing said message-related data.

7. The method of claim 1, said first step further including the step of transforming (602) a coordinate system expressing said message-related data into a coordinate system with coordinate axes having different weights added thereto.

8. A scalar multiplication method for calculating a scalar-multiplied point from a scalar value and a point on an elliptic curve in an elliptic curve cryptosystem, comprising the steps of:

randomizing (502, 602, 1203) said point on said elliptic curve; and
executing (504, 514, 515, 517, 518, 613, 614, 1204, 1209, 1210) an operation upon a value derived from said randomized point and a value derived from said point on said elliptic curve without randomizing said point on said elliptic curve.

9. The method of claim 8, further comprising the steps of:

judging (513, 615) a value of each bit of said scalar value;
carrying out (514, 517, 616, 617) an operation upon each bit of said scalar value in accordance with a result of said judgement; and
combining (514, 517, 614) results of said operations;

wherein said step of carrying out an operation upon each bit includes calculating steps (514-519, 616, 617) independent of said judged value of bits.

10. The method of claim 9, wherein said step of carrying out (1209, 1210) an operation upon each bit is executed a predetermined number of times independent of bit length of said scalar value.

11. The method of claim 8, wherein one of the following is used as said elliptic curve:

a Montgomery-form elliptic curve,
a Weierstrass-form elliptic curve,
an elliptic curve defined on a finite field with characteristics 2,
an elliptic curve defined on an OEF.

12. A data generation method comprising the step of calculating (515, 518, 613, 1209) scalar-multiplication of data obtained by said operation executing step in a scalar multiplication method according to claim 8.

13. A signature generation method for generating signature data from data, comprising the step of sending (512, 621, 1213) data obtained by said operation executing step in a scalar multiplication method according to claim 8, as signature data.

14. A decryption method comprising the step of generating (521, 621, 1213) decrypted data from encrypted data by said operation executing step in a scalar multiplication method according to claim 8.

15. A scalar multiplication system (115, 135, 202, 715) for calculating a scalar-multiplied point from a scalar value and a point on an elliptic curve in an elliptic curve cryptosystem, comprising:

a randomizing portion (402) for randomizing said point on said elliptic curve; and
an operating portion (403, 404) for executing an operation upon a value derived from said randomized point

and a value derived from said point on said elliptic curve without randomization, so as to calculate said scalar-multiplied point.

16. A signature generation system (701) comprising:

a signature portion (712) for generating signature data from message data; and
a scalar multiplication portion (715) for calculating scalar multiplication in response to a request from said signature portion;

wherein said scalar multiplication portion calculates said scalar multiplication by use of a scalar multiplication method according to claim 8.

17. A decryption system (121) comprising:

a decryption portion (132, 201) for generating decrypted data from encrypted data; and
a scalar multiplication portion (135, 202) for calculating scalar multiplication in response to a request from said decryption portion;

wherein said scalar multiplication portion calculates said scalar multiplication by use of a scalar multiplication method according to claim 8, and wherein said encrypted data and a scalar multiplication result are operated to obtain decrypted data.

18. A storage medium (102, 122, 203, 702, 722) storing programs concerned with a scalar multiplication method according to claim 8 or claim 11, or programs concerned with a data generation method according to claim 12.

FIG.1

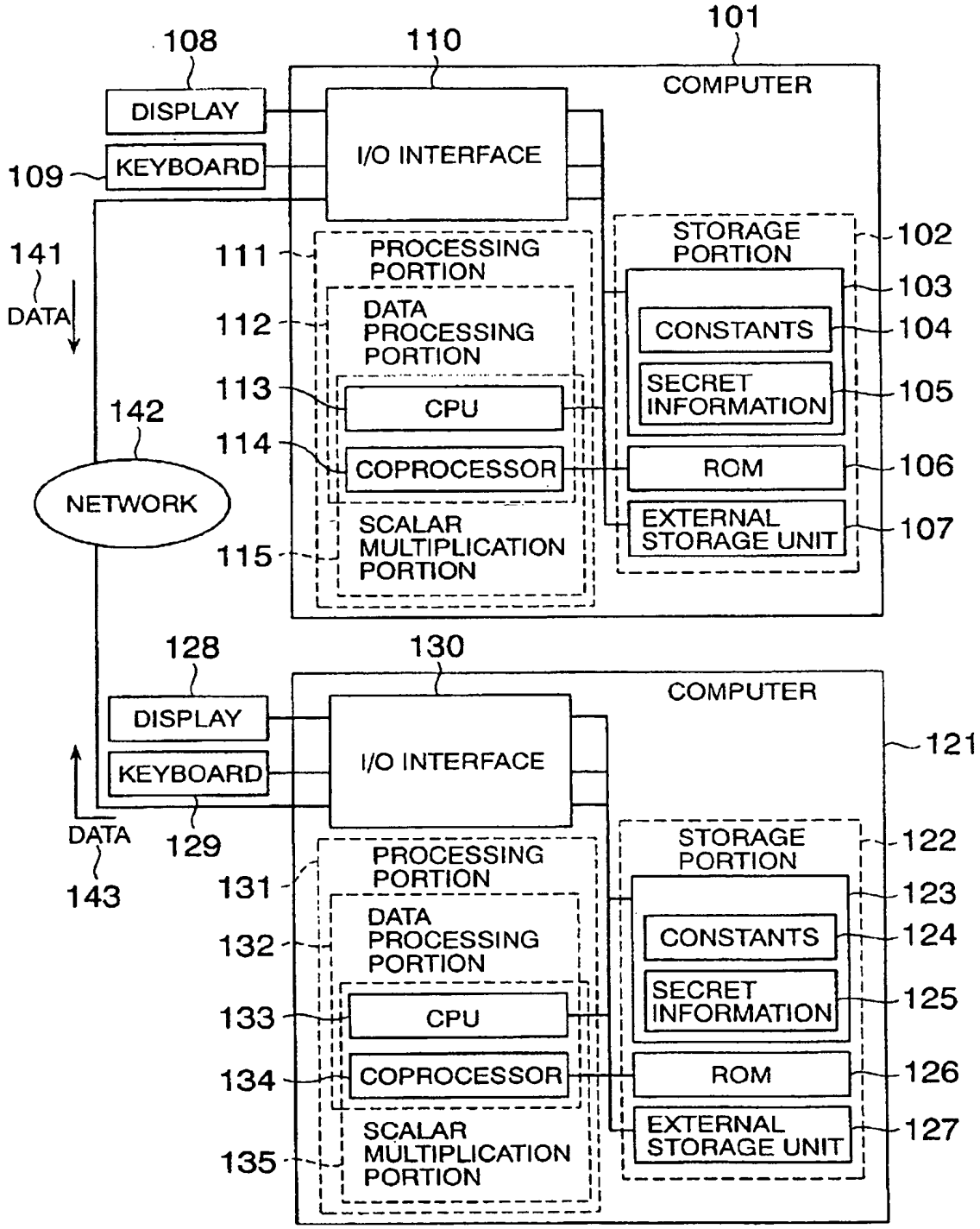

101
108
110
COMPUTER
DISPLAY
I/O INTERFACE
109
KEYBOARD
141
DATA
111
PROCESSING PORTION
STORAGE PORTION
102
112
DATA PROCESSING PORTION
103
CONSTANTS
104
113
CPU
SECRET INFORMATION
105
142
114
COPROCESSOR
ROM
106
NETWORK
EXTERNAL STORAGE UNIT
107
115
SCALAR MULTIPLICATION PORTION
128
130
COMPUTER
121
DISPLAY
I/O INTERFACE
KEYBOARD
DATA
129
143
122
PROCESSING PORTION
STORAGE PORTION
131
123
DATA PROCESSING PORTION
132
CONSTANTS
124
SECRET INFORMATION
125
133
CPU
134
COPROCESSOR
ROM
126
EXTERNAL STORAGE UNIT
127
135
SCALAR MULTIPLICATION PORTION

FIG.2

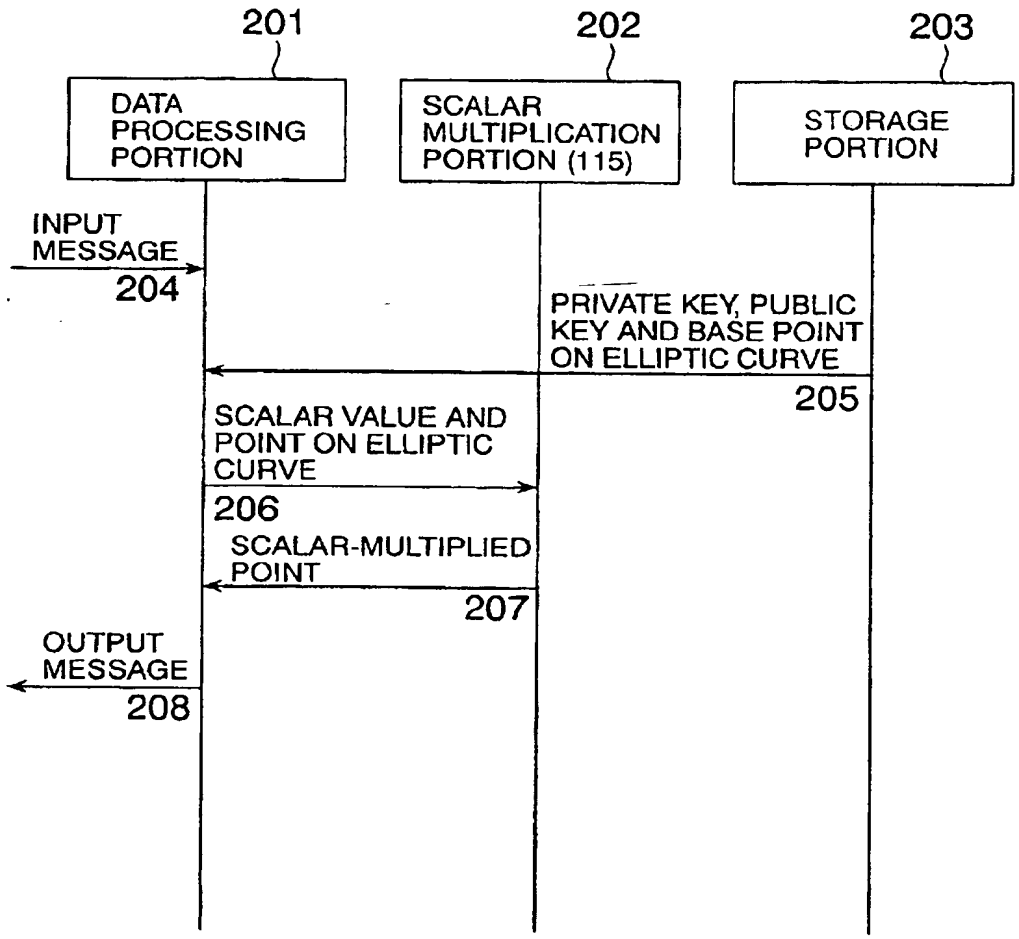

201
202
203
DATA PROCESSING PORTION
SCALAR MULTIPLICATION PORTION (115)
STORAGE PORTION
INPUT MESSAGE
204
PRIVATE KEY, PUBLIC KEY AND BASE POINT ON ELLIPTIC CURVE
205
SCALAR VALUE AND POINT ON ELLIPTIC CURVE
206
SCALAR-MULTIPLIED POINT
207
OUTPUT MESSAGE
208

FIG.3

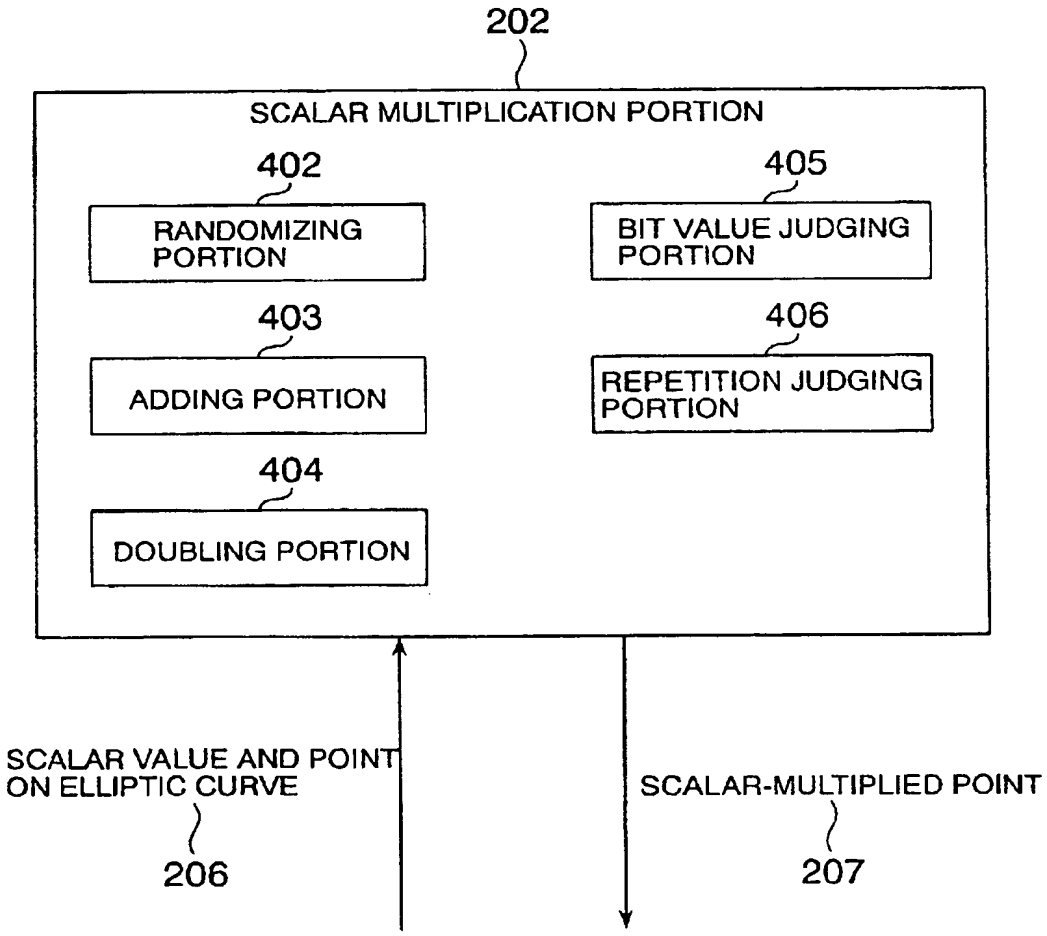

202
SCALAR MULTIPLICATION PORTION
402
RANDOMIZING PORTION
403
ADDING PORTION
404
DOUBLING PORTION
405
BIT VALUE JUDGING PORTION
406
REPETITION JUDGING PORTION
SCALAR VALUE AND POINT ON ELLIPTIC CURVE
206
SCALAR-MULTIPLIED POINT
207

FIG.4

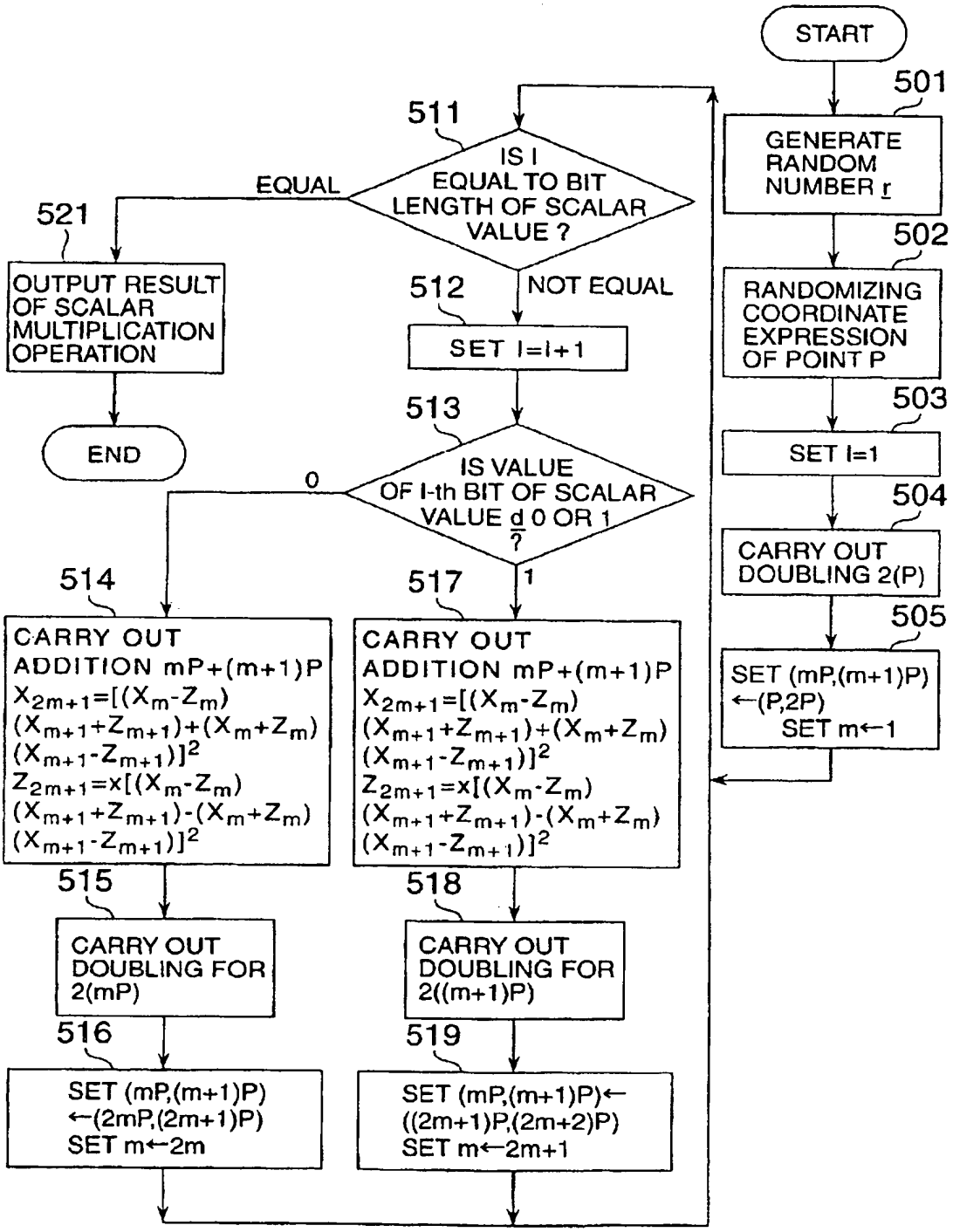

START
501
GENERATE RANDOM NUMBER r
502
RANDOMIZING COORDINATE EXPRESSION OF POINT P
503
SET I=1
504
CARRY OUT DOUBLING 2(P)
505
SET (mP,(m+1)P) ←(P,2P) SET m←1
511
IS I EQUAL TO BIT LENGTH OF SCALAR VALUE ?
EQUAL
NOT EQUAL
521
OUTPUT RESULT OF SCALAR MULTIPLICATION OPERATION
END
512
SET I=I+1
513
IS VALUE OF I-th BIT OF SCALAR VALUE d 0 OR 1 ?
0
1
514
CARRY OUT ADDITION mP+(m+1)P
$X_{2m+1}=[(X_m-Z_m)(X_{m+1}+Z_{m+1})+(X_m+Z_m)(X_{m+1}-Z_{m+1})]^2$
$Z_{2m+1}=x[(X_m-Z_m)(X_{m+1}+Z_{m+1})-(X_m+Z_m)(X_{m+1}-Z_{m+1})]^2$
515
CARRY OUT DOUBLING FOR 2(mP)
516
SET (mP,(m+1)P) ←(2mP,(2m+1)P) SET m←2m
517
CARRY OUT ADDITION mP+(m+1)P
$X_{2m+1}=[(X_m-Z_m)(X_{m+1}+Z_{m+1})+(X_m+Z_m)(X_{m+1}-Z_{m+1})]^2$
$Z_{2m+1}=x[(X_m-Z_m)(X_{m+1}+Z_{m+1})-(X_m+Z_m)(X_{m+1}-Z_{m+1})]^2$
518
CARRY OUT DOUBLING FOR 2((m+1)P)
519
SET (mP,(m+1)P)← ((2m+1)P,(2m+2)P) SET m←2m+1

FIG.5

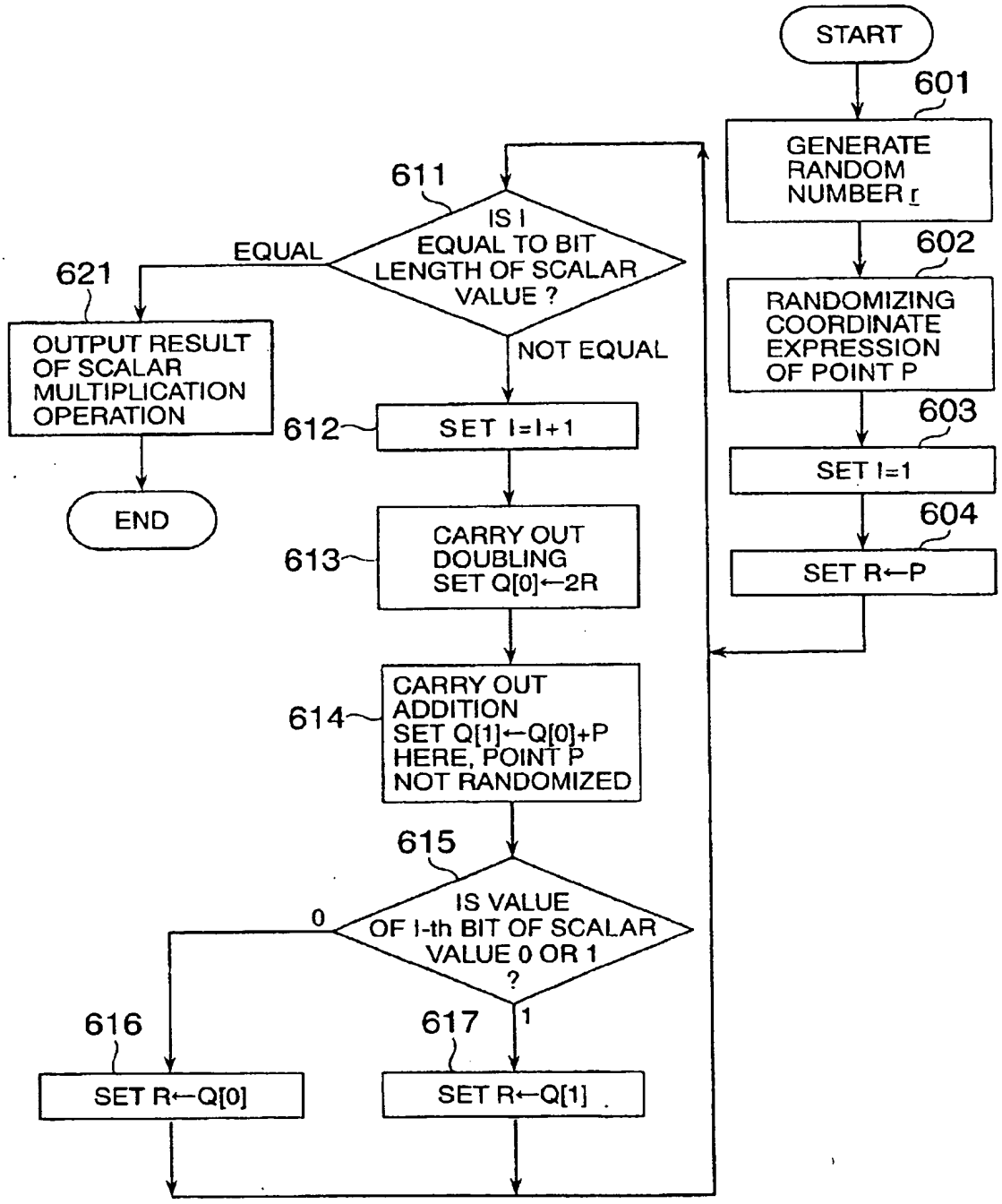

START
601
GENERATE RANDOM NUMBER r
602
RANDOMIZING COORDINATE EXPRESSION OF POINT P
603
SET I=1
604
SET R←P
611
IS I EQUAL TO BIT LENGTH OF SCALAR VALUE ?
EQUAL
NOT EQUAL
621
OUTPUT RESULT OF SCALAR MULTIPLICATION OPERATION
END
612
SET I=I+1
613
CARRY OUT DOUBLING SET Q[0]←2R
614
CARRY OUT ADDITION SET Q[1]←Q[0]+P HERE, POINT P NOT RANDOMIZED
615
IS VALUE OF I-th BIT OF SCALAR VALUE 0 OR 1 ?
0
1
616
SET R←Q[0]
617
SET R←Q[1]

FIG.6

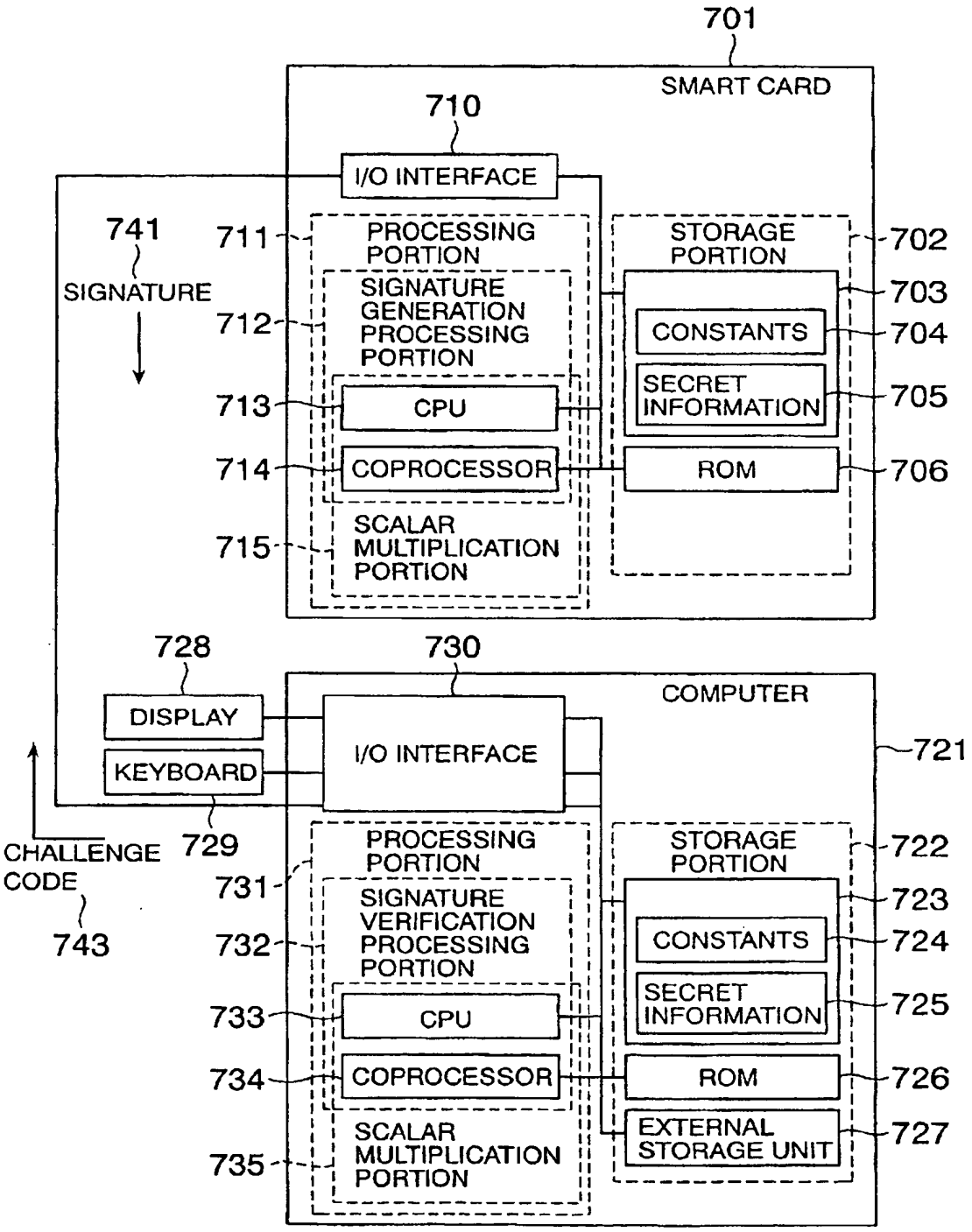
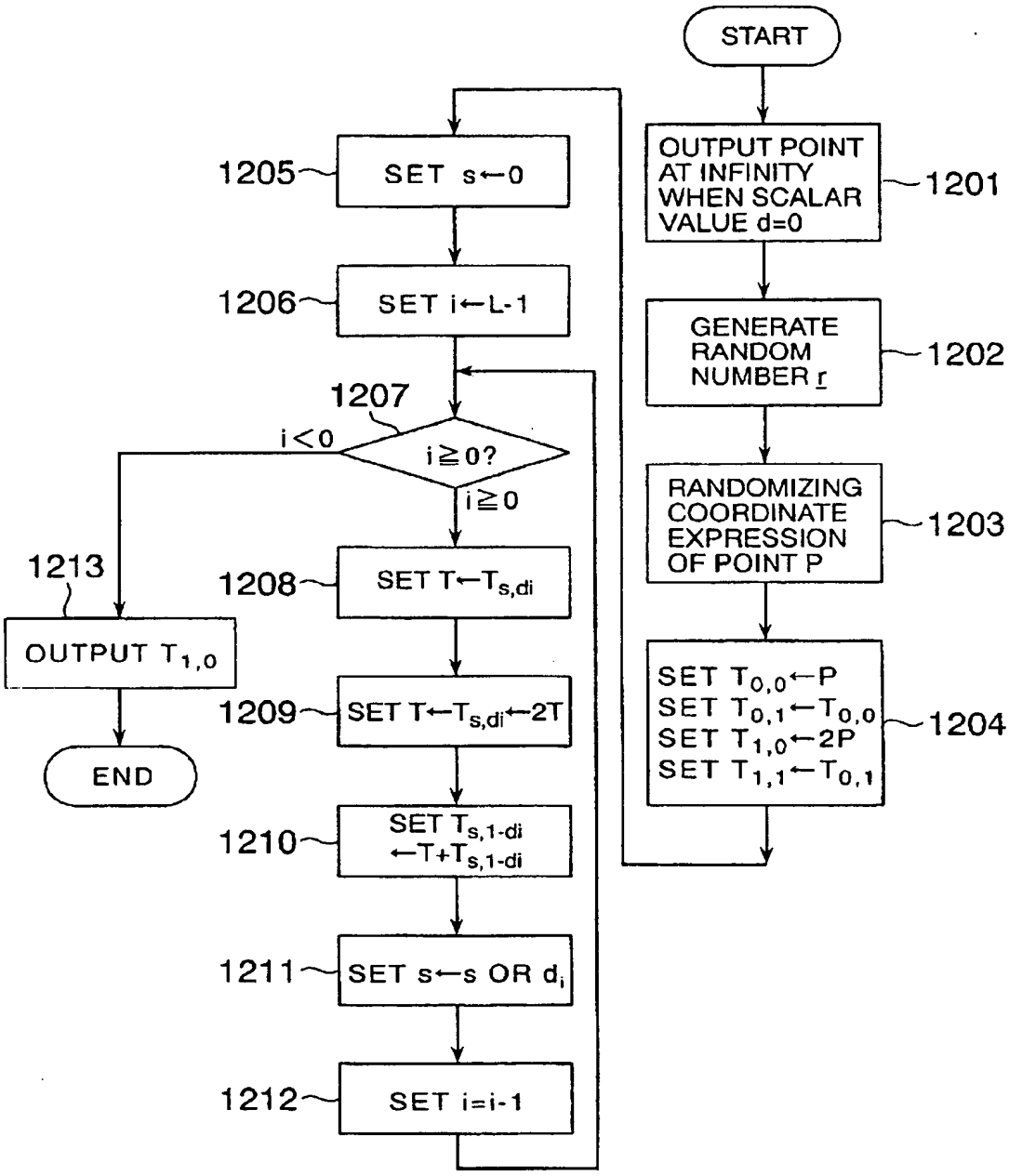

701
SMART CARD
710
I/O INTERFACE
741
SIGNATURE
711
PROCESSING PORTION
712
SIGNATURE GENERATION PROCESSING PORTION
713
CPU
714
COPROCESSOR
715
SCALAR MULTIPLICATION PORTION
STORAGE PORTION
702
703
CONSTANTS
704
SECRET INFORMATION
705
ROM
706
728
DISPLAY
KEYBOARD
729
730
COMPUTER
I/O INTERFACE
721
CHALLENGE CODE
743
PROCESSING PORTION
731
SIGNATURE VERIFICATION PROCESSING PORTION
732
733
CPU
734
COPROCESSOR
735
SCALAR MULTIPLICATION PORTION
STORAGE PORTION
722
723
CONSTANTS
724
SECRET INFORMATION
725
ROM
726
EXTERNAL STORAGE UNIT
727

FIG.7


START
OUTPUT POINT AT INFINITY WHEN SCALAR VALUE d=0
1201
GENERATE RANDOM NUMBER r
1202
RANDOMIZING COORDINATE EXPRESSION OF POINT P
1203
SET $T_{0,0} \leftarrow P$
SET $T_{0,1} \leftarrow T_{0,0}$
SET $T_{1,0} \leftarrow 2P$
SET $T_{1,1} \leftarrow T_{0,1}$
1204
1205
SET s←0
1206
SET i←L-1
1207
i<0
i≧0?
i≧0
1208
SET $T \leftarrow T_{s,di}$
1209
SET $T \leftarrow T_{s,di} \leftarrow 2T$
1210
SET $T_{s,1-di} \leftarrow T+T_{s,1-di}$
1211
SET $s \leftarrow s$ OR d_i
1212
SET i=i-1
1213
OUTPUT $T_{1,0}$
END

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 01 5805

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
P,X	OKEYA K ; MIYAZAKI K ; SAKURAI K : "A fast scalar multiplication method with randomized projective coordinates on a Montgomery-form elliptic curve secure against side channel attacks" INFORMATION SECURITY AND CRYPTOLOGY - ICISC 2001. 4TH INTERNATIONAL CONFERENCE. PROCEEDINGS (LECTURE NOTES IN COMPUTER SCIENCE VOL.2288), SPRINGER-VERLAG, 7 December 2001 (2001-12-07), pages 428-439, XP002218623 Seoul, South Korea ISBN: 3-540-43319-8 * the whole document * ---	1-18	G06F7/72 H04L9/30
D,A	OKEYA K ; SAKURAI K : "Power analysis breaks elliptic curve cryptosystems even secure against the timing attack " PROGRESS IN CRYPTOLOGY - INDOCRYPT 2000. FIRST INTERNATIONAL CONFERENCE IN CRYPTOLOGY IN INDIA. PROCEEDINGS (LECTURE NOTES IN COMPUTER SCIENCE VOL.1977), SPRINGER-VERLAG, 13 December 2000 (2000-12-13), pages 178-190, XP002218624 Calcutta, India ISBN: 3-540-41452-5 * page 178 - page 186 * ---	1-18	

-/--

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

G06F
H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	28 October 2002	Carnerero Álvaro, F

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 01 5805

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
D,A	OKEYA K ET AL: "EFFICIENT ELLIPTIC CURVE CRYPTOSYSTEMS FROM A SCALAR MULTIPLICATION ALGORITHM WITH RECOVERY OF THE GAMMA-COORDINATE ON A MONTGOMERY-FORM ELLIPTIC CURVE" CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS. 3RD INTERNATIONAL WORKSHOP, CHES 2001, PARIS, FRANCCE, MAY 14 - 16, 2001 PROCEEDINGS, LECTURE NOTES IN COMPUTER SCIENCE, BERLIN: SPRINGER, DE, vol. 2162, 14 May 2001 (2001-05-14), pages 126-141, XP001061163 ISBN: 3-540-42521-7 * page 126 - page 133 *	1-18	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	28 October 2002	Carnerero Álvaro, F

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)